Examining ASP.NET's Membership, Roles, and Profile - Part 4

By Scott Mitchell

Introduction

The ASP.NET `Membership` class provides a `ValidateUser(userName, password)` method that returns a Boolean value indicating whether or not a user's supplied credentials are valid. This method is automatically utilized from the Login Web control and can also be used programmatically, if needed. In the Membership system, there are multiple scenarios by which a user's credentials can be invalid:

- The username supplied might not exist in the membership directory
- The username may exist, but the supplied password might be incorrect
- The username and password may be correct, but:
  - The user may not yet be approved
  - The user may be locked out; this can happen if the user attempts to login with an invalid password for a specified number of tries (five, by default)

Unfortunately, the `ValidateUser(userName, password)` method just returns False if the credentials are invalid, and does not include information as to why, exactly, the credentials are invalid. For the Login control, when `ValidateUser(userName, password)` returns False the message, "Your login attempt was not successful. Please try again." is displayed, by default. If a user is locked out or their account not yet approved, such a message - which will be shown even in the face of the correct username and password - can easily lead to a confused and frustrated user.

In this article we'll see how to provide additional feedback during the login process to help alleviate any such confusion. Moreover, we'll see how to audit invalid logins and present the data in a report. Read on to learn more!

Approved and Locked Out User Accounts

The user accounts in the ASP.NET Membership system can be accessed and modified programmatically through the `Membership` and `MembershipUser` classes. `Membership` contains a bevy of static methods that offer the ability to retrieve information about all users, a particular user, to update a user, and so on, while the `MembershipUser` class contains properties that describe the state of a specific user (`UserName`, `Email`, `LastLoggedOnDate`, and so on).

The Membership system enables user accounts to be marked as inactive (not approved) and locked out. When creating a new user account, the account is, by default, approved. However, in some scenarios you might want to have an administrator manually approve new accounts before they become active, or have the user progress through some automated process (like clicking on a verification link sent through email). In either case, the newly created user would be marked as inactive. Such a user cannot log in to the site, as the `ValidateUser(userName, password)` method will always return False, regardless of whether they entered their correct credentials.

Since the process of authenticating through a forms-based scheme involves simply sending the user's credentials over an HTTP request, an attacker could attempt to break into a user's account by writing a script that looped through a dictionary of common passwords, sending an appropriately formatted HTTP request for a particular user for each password in the dictionary. To help stop such attacks, the

Membership system automatically locks out a user if a certain number of invalid password attempts have transpired in a specified window of time. These settings default to five invalid password attmepts within a ten minute window, but can be customized in `Web.config` if needed (refer to Part 1 of this article series for information on customizing the membership provider). As with unapproved users, a locked user cannot log in to the website regardless of whether or not they provide their credentials. In order to unlock a user account, the `MembershipUser` class's `UnlockUser()` method must be invoked.

When attempting to login throught the Login Web control, a user will see the same message whether they are unable to log in due to an invalid username, an invalid password, or because their account has not been approved or is currently locked out. Rather than showing the user the same blanket message, we can customize the login page to display a more appropriate message.

Displaying an Informative Message in the Face of an Invalid Login

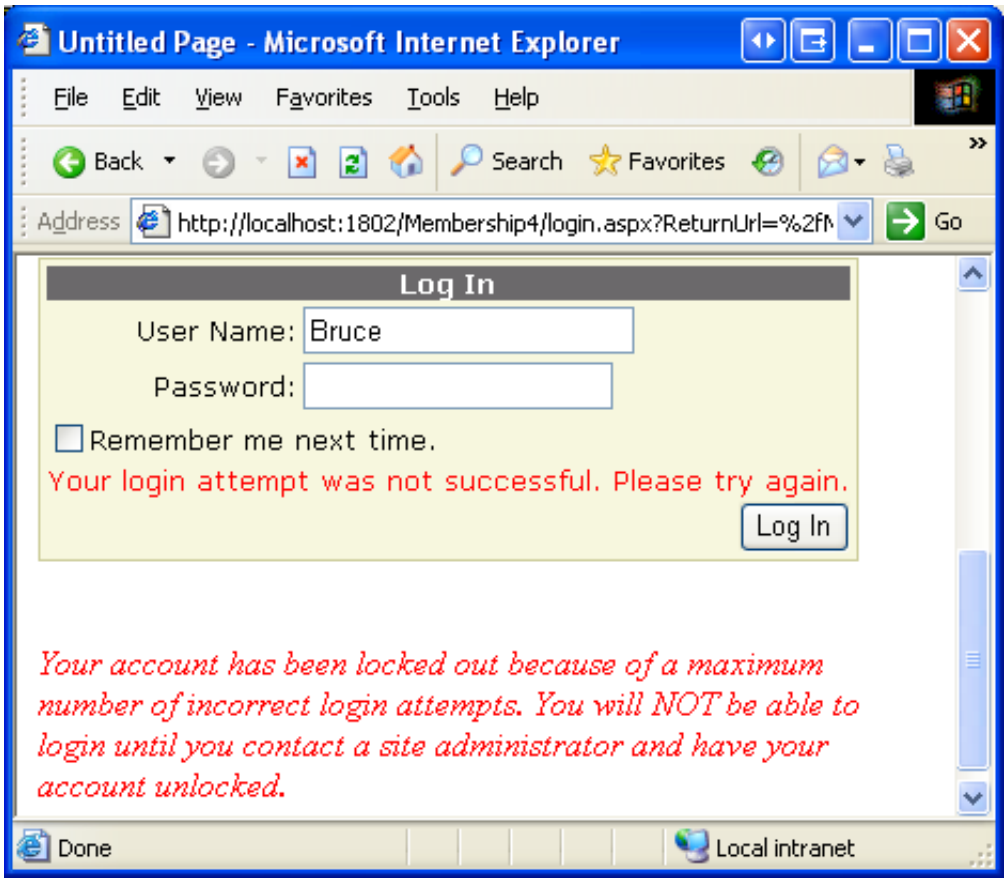
When a user attempts to log on to the website using the Login Web control, the Login control's `LoginError` event fires. This event handler is not passed any information that explains why the login failed; however, we can get the username and password the user attempted to use via the Login control's `UserName` and `Password` properties. Using the `UserName` property, we can get information about the user account through the `Membership.GetUser(userName)` method. This method returns a `MembershipUser` object, from which we can check the `IsApproved` and `IsLockedOut` properties to determine why the user's credentials were deemed invalid.

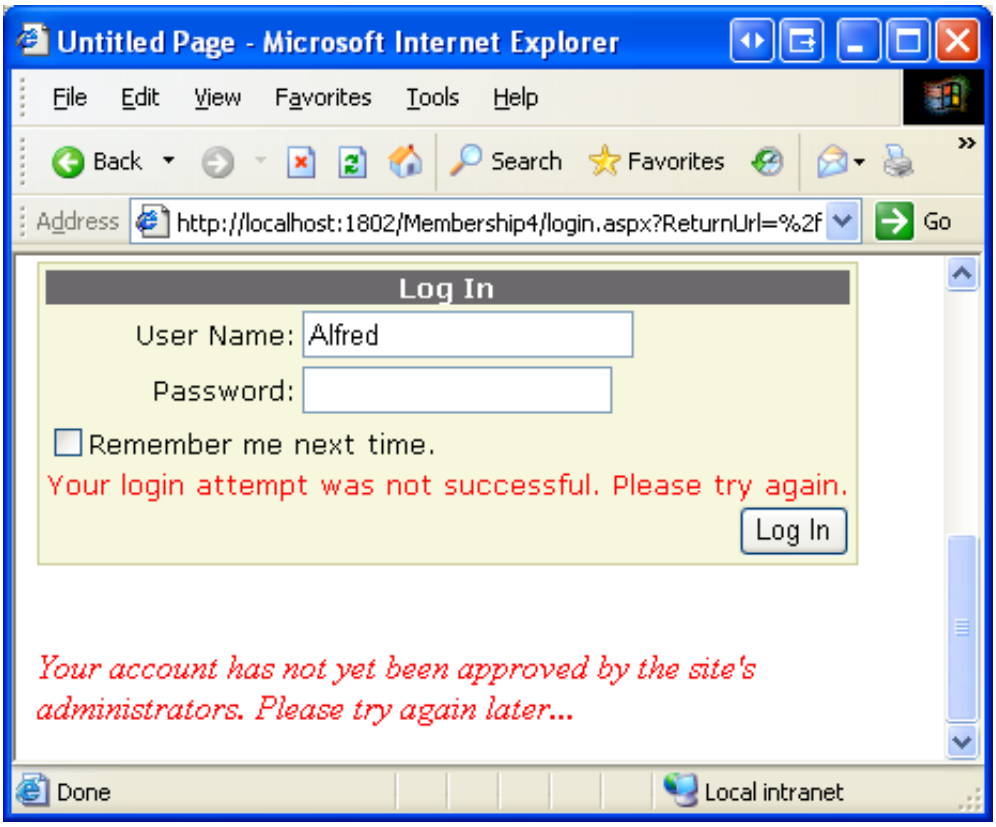
The following event handler code shows how to accomplish this. The resulting help message is displayed in the Label Web control named `LoginErrorDetails`; you can view the Login page's complete code and declarative markup by downloading the complete code at the end of this article.

```
Protected Sub Login1_LoginError(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Login1.LoginError
    'There was a problem logging in the user

    'See if this user exists in the database
    Dim userInfo As MembershipUser = Membership.GetUser(Login1.UserName)

    If userInfo Is Nothing Then
        'The user entered an invalid username...
        LoginErrorDetails.Text = "There is no user in the database with the username " &
Login1.UserName
    Else
        'See if the user is locked out or not approved
        If Not userInfo.IsApproved Then
            LoginErrorDetails.Text = "Your account has not yet been approved by the
site's administrators. Please try again later..."
        ElseIf userInfo.IsLockedOut Then
            LoginErrorDetails.Text = "Your account has been locked out because of a
maximum number of incorrect login attempts. You will NOT be able to login until you
contact a site administrator and have your account unlocked."
        Else
            'The password was incorrect (don't show anything, the Login control already
describes the problem)
            LoginErrorDetails.Text = String.Empty
        End If
    End If
End Sub
```

With this code, the user will see a more informative message if their login fails. The following screenshots show the results when attempting to login with Bruce (whose account has been locked out) and Alfred (whose account has yet to be approved). Without the above event handler, these users would have just

seen the standard "Your login attempt was not successful. Please try again." message when attempting to login, even had they entered the correct credentials (due to their locked out / approved status). (Clearly this would be confusing to Bruce and Alfred, who might not realize that their account has yet to been approved or has been locked out.)





Logging Invalid Login Attempts

While the ASP.NET Membership system keeps track of invalid password attempts and locks out a user's account if a specified threshold is surpassed, it doesn't log any of the invalid login attempts. Such a log can provide a quick report to see what users have had troubles logging in, what users are logged out, and which ones have yet to be approved. This data can also help in a security audit, identifying patterns that might be attackers who are relying on some sort of dictionary attack.

To capture such information, I created a database table named `InvalidCredentialsLog` in the membership database with the following schema:

InvalidCredentialsLog		
Column	**Data Type**	**Comments**
`InvalidCredentialsLogID`	`int, PK, IDENTITY, NOT NULL`	Uniquely identifies each record
`UserName`	`nvarchar(256), NOT NULL`	The name entered by the user, when logging in
`Password`	`nvarchar(128)`	The password attempted by the user; only recorded if the user enters an *incorrect* password or the username supplied does not exist in the database (i.e., does not appear for users entering valid credentials, but who are locked out or not approved)
`IsApproved`	`bit`	Whether or not the user account is approved
`IsLockedOut`	`bit`	Whether or not the user account is locked out
`IPAddress`	`varchar(15)`	The IP Address of the user who supplied invalid credentials
`LoginAttemptDate`	`datetime, NOT NULL`	The date/time the invalid login attempt occurred (defaults to `getdate()`)

The Membership system includes an `ApplicationID`, which allows multiple applications to store their user account information in a single database. Ideally, this table would include the `ApplicationID` and the report would only show those invalid credentials for the current application (assuming you use a single membership store for multiple applications). I leave adding this feature as an exercise for the reader!

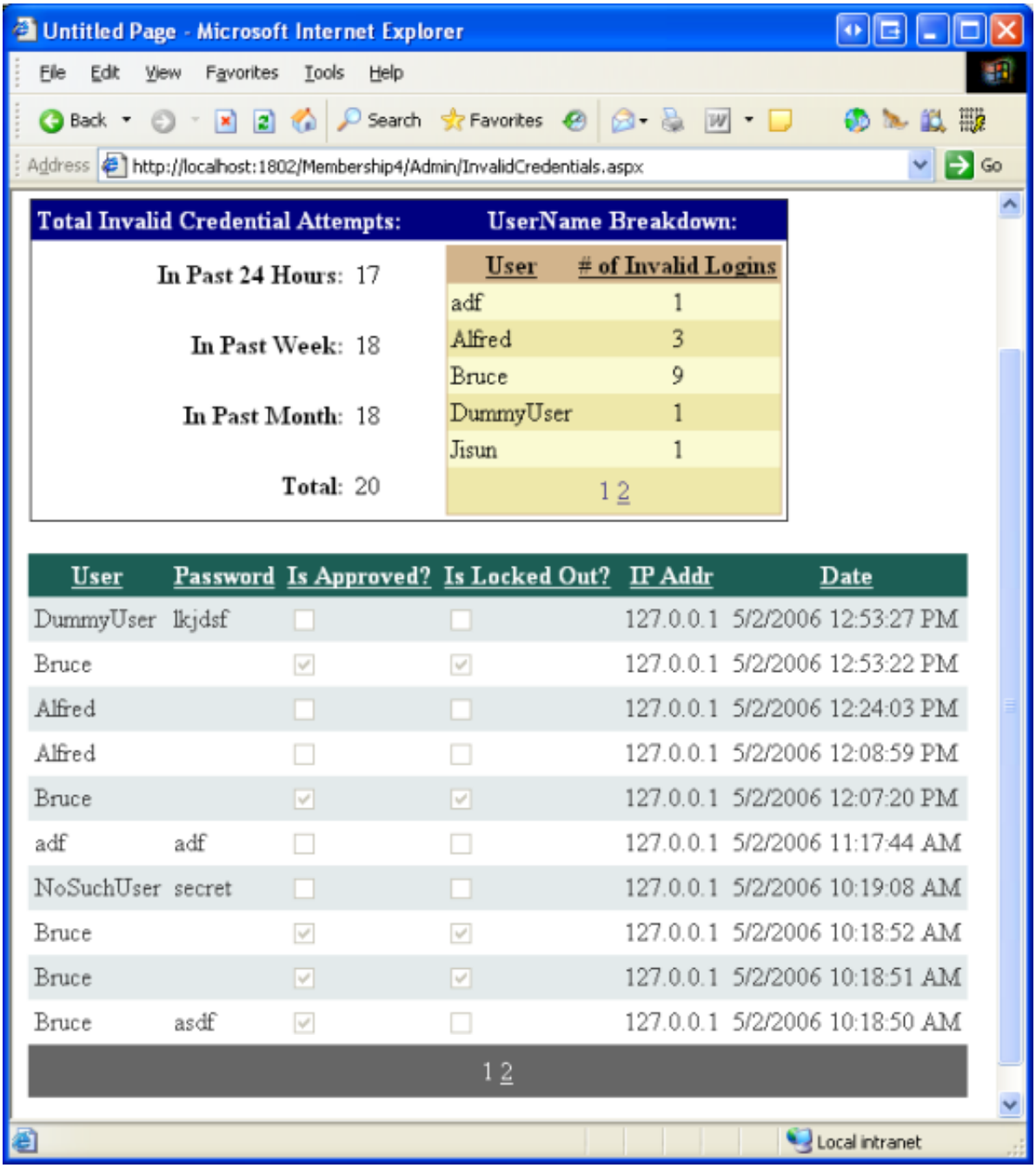
Next, I created a stored procedure - `InvalidCredentialsLog_Insert` - that takes in the user's username, password, and IP address. It then checks to see if the username maps to a user in the `aspnet_Users` table and, if so, grabs the user's `IsApproved` and `IsLockedOut` fields values. It then `INSERT`s this information into the `InvalidCredentialsLog_Insert` table.

When the user enters invalid credentials in the login page, this stored procedure needs to be called, passing in the user's information. To accomplish this I created a SqlDataSource (`InvalidCredentialsLogDataSource`) and configured it to call the stored procedure. Then, I extended the Login Web control's `LoginError` event handler to set the parameters for this stored procedure and invoke it:

```
Protected Sub Login1_LoginError(ByVal sender As Object, ByVal e As System.EventArgs)
```

```
Handles Login1.LoginError
    'Set the parameters for InvalidCredentialsLogDataSource
    InvalidCredentialsLogDataSource.InsertParameters("ApplicationName").DefaultValue =
Membership.ApplicationName
    InvalidCredentialsLogDataSource.InsertParameters("UserName").DefaultValue =
Login1.UserName
    InvalidCredentialsLogDataSource.InsertParameters("IPAddress").DefaultValue =
Request.UserHostAddress

    'The password is only supplied if the user enters an invalid username or invalid
password - set it to Nothing, by default
    InvalidCredentialsLogDataSource.InsertParameters("Password").DefaultValue = Nothing


    'There was a problem logging in the user
    'See if this user exists in the database
    Dim userInfo As MembershipUser = Membership.GetUser(Login1.UserName)
    If userInfo Is Nothing Then
        'The user entered an invalid username...
        LoginErrorDetails.Text = "There is no user in the database with the username " &
Login1.UserName

        'The password is only supplied if the user enters an invalid username or invalid
password
        InvalidCredentialsLogDataSource.InsertParameters("Password").DefaultValue =
Login1.Password
    Else
        'See if the user is locked out or not approved
        If Not userInfo.IsApproved Then
            LoginErrorDetails.Text = "Your account has not yet been approved by the
site's administrators. Please try again later..."
        ElseIf userInfo.IsLockedOut Then
            LoginErrorDetails.Text = "Your account has been locked out because of a
maximum number of incorrect login attempts. You will NOT be able to login until you
contact a site administrator and have your account unlocked."
        Else
            'The password was incorrect (don't show anything, the Login control already
describes the problem)
            LoginErrorDetails.Text = String.Empty

            'The password is only supplied if the user enters an invalid username or
invalid password
            InvalidCredentialsLogDataSource.InsertParameters("Password").DefaultValue =
Login1.Password
        End If
    End If

    'Add a new record to the InvalidCredentialsLog table
    InvalidCredentialsLogDataSource.Insert()
End Sub
```

In addition to this stored procedure and event handler, I build a simple report page that shows *all* of the invalid credentials in a pageable, sortable GridView, along with summary information, as shown in the screen shot below. This report page, along with the stored procedure and a working demo, can be downloaded at the end of this article.



For popular websites, the `InvalidCredentialsLog` table could grow very large. It might make sense to instigate some policy that involved deleting invalid entries older than a certain date (such as a SQL Server Job that ran weekly, deleting the audit history older than, say, three months). The report shown in the screen shot above is very simple and has not been optimized for working with large result sets. It currently uses default paging, bringing back *every* record from the database for each page of data shown. For significantly large `InvalidCredentialsLog` tables this could introduce a less than ideal page load time. Consider upgrading the paging logic used here to use custom paging, which intelligently grabs only those records needed for displaying the current page of data; see Custom Paging in ASP.NET 2.0 with SQL Server 2005 for more information.

Conclusion

In this fourth part of the Membership, Roles, and Profile article series, we saw how to enhance the login process by including more descriptive information for those users attempting to login whose accounts are not yet approved or have been locked out. This level of extensibility is due in part to the Membership API, which can be accessed programmatically, declaratively (through data source controls), and through

Web controls (such as the Login Web control). The audit table presented in this article could be expanded to capture not just invalid logins, but also valid ones.

Be sure to check out the download available at the end of this article. It includes the complete source code and database additions discussed throughout this article. Additionally, it includes an Admin page that provides a GridView that lists the users in the system and allows the user to quickly toggle their approved status, to unlock locked out users, and to toggle whether or not a given user is in the Administrator role. (Only users in the Administrator role can view the Admin-related web pages).

Happy Programming!

- By Scott Mitchell

Attachments

- Download the code used in this article

Article Information	
Article Title:	ASP.NET.Examining ASP.NET's Membership, Roles, and Profile - Part 4
Article Author:	Scott Mitchell
Published Date:	May 3, 2006
Article URL:	http://www.4GuysFromRolla.com/articles/050306-1.aspx